UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

William da Silva Wiethan

# UTILIZAÇÃO DE APRENDIZADO PROFUNDO PARA DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES

465
Santa Maria, RS
2019

**William da Silva Wiethan**

# UTILIZAÇÃO DE APRENDIZADO PROFUNDO PARA DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADOR: Prof. João Vicente Ferreira Lima

465
Santa Maria, RS
2019

**William da Silva Wiethan**

# UTILIZAÇÃO DE APRENDIZADO PROFUNDO PARA DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

**Aprovado em 4 de dezembro de 2019:**

**João Vicente Ferreira Lima, Dr. (UFSM)**
(Orientador)

**Celio Trois, Dr. (UFSM)**

**João Carlos Damasceno Lima, Dr. (UFSM)**

Santa Maria, RS
2019

# RESUMO

## UTILIZAÇÃO DE APRENDIZADO PROFUNDO PARA DETECÇÃO DE ANOMALIAS EM REDES DE COMPUTADORES

AUTOR: William da Silva Wiethan
ORIENTADOR: João Vicente Ferreira Lima

A ideia de uma máquina de aprendizagem inteligente e independente fascinou os humanos por décadas, hoje, diversos fatores se uniram para tornar a aprendizagem de máquina uma realidade. Neste contexto da inteligência artificial o aprendizado profundo tem sido uma das ferramentas de aprendizado de máquina mais utilizadas, contudo sua utilização em detecção de anomalias em redes, uma área relevante, ainda não é tão explorada se compararmos com áreas como reconhecimento de imagens e identificação de voz. Nesse trabalho foi feito um estudo sobre a utilização de aprendizado profundo na detecção de anomalias em redes, através da implementação e análise de três modelos de classificadores baseados no aprendizado profundo, que foram testados em execuções de validação cruzada e em dados de uma base de dados paralela a base de seu treinamento. Como resultado pode-se dizer que o classificador binário, teve um desempenho razoável em relação aos demais classificadores propostos no trabalho.

**Palavras-chave:** aprendizado de máquina. aprendizado profundo. detecção de intrusão.

# ABSTRACT

## USING DEEP LEARNING TO DETECT ANOMALIES IN COMPUTER NETWORKS

AUTHOR: William da Silva Wiethan
ADVISOR: João Vicente Ferreira Lima

The idea of an intelligent and independent learning machine has fascinated humans for decades, today several factors have come together to make machine learning a reality. In this context of artificial intelligence, deep learning has been one of the most widely used machine learning tools, but its use in network anomaly detection, a relevant area, is still not explored if we compare with areas such as image recognition and voice identification. In this work we study the use of deep learning in detecting anomalies in networks by implementing and analyzing three models of deep learning-based classifiers, which were tested in cross-validation runs and in data from a database parallel to the one of its training. As a result, it can be said that the binary classifier, had a reasonable performance in relation to the other classifiers proposed in the work.



**Keywords:** machine learning. deep learning. intrusion detected.

## LISTA DE FIGURAS

## LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

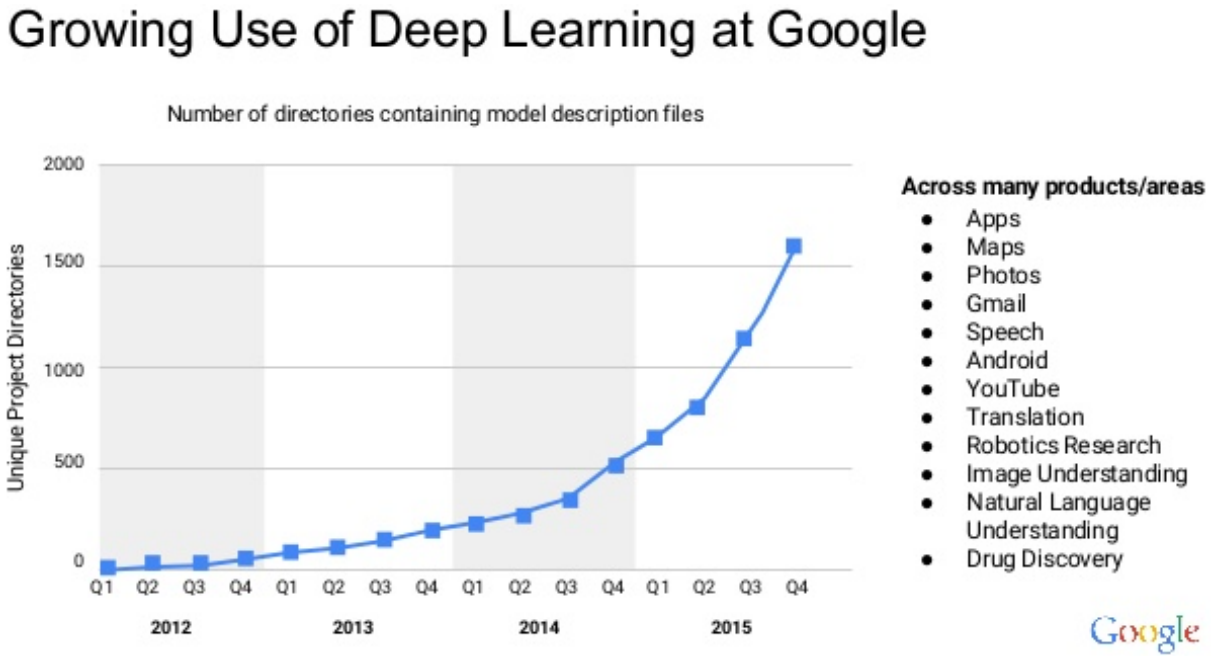
| | |
|---|---|
| $CPU$ | Central Processing Unit (Unidade Central de Processamento) |
| $DL$ | Aprendizado Profundo (Deep Learning) |
| $GPU$ | Graphics Processing Unit (Unidade de Processamento Gráfico) |
| $IA$ | Inteligência Artificial |
| $IDS$ | Intrusion Detection System (Sistemas de Detecção de Intrusão) |
| $IP$ | Internet Protocol |
| $ML$ | Aprendizado de Máquina (*Machine Learning*) |
| $MLN$ | Machine Learning for Networking (Aprendizado de Máquina para Rede) |
| $RN$ | Rede Neural |

**SUMÁRIO**

# 1 INTRODUÇÃO

As redes de computadores e principalmente a Internet se tornaram indispensáveis para a vida moderna, sendo utilizadas para várias finalidades e pelos mais diversos nichos, profissional, comercial, social, educacional, entre outros (MARZ; WARREN, 2015). É importante identificar comportamentos anômalos e ataques de usuários maliciosos a fim de manter a integridade e o adequado funcionamento das redes, porém a quantidade massiva de dados gerada pelos fluxos de tráfego da rede torna complicada uma análise manual de possíveis ataques, de modo que se faz surgir a necessidade de ferramentas inteligentes que ajudem na classificação e identificação de potenciais riscos. Ferramentas como TensorFlow, Keras e Scikt Learning, potencializam o processo de criação de modelos de Aprendizagem Profunda (*Deep Learning*, DL), pois facilitam a implementação dos algoritmos, mas segundo (FADLULLAH et al., 2017), os estudos e implementações feitos até o momento com a técnica de aprendizado profundo são mais voltados para outras áreas, como reconhecimento de imagens e voz.

Figura 1.1 – Crescimento do aprendizado profundo.




Fonte: Google (ttps://www.google.com/)

Aprendizado profundo é na atualidade a principal tecnologia utilizada em sistemas de Inteligência Artificial, a Figura 1.1 nos da uma amostra do interesse no aprendizado profundo ao longo dos anos. Redes Neurais Profundas, que tentam simular cérebro humano em máquinas, vêm sendo usadas com sucesso em atividades de visão computacional e reconhecimento da fala, com resultados cada vez mais próximos do real. Esta subárea da Inteligência Artificial (IA) vem crescendo rapidamente, devido a melhorias de desem-

penho em hardware e uso de Unidades de Processamento Gráfico (*Graphics Processing Unit*, GPU). Ferramentas e *frameworks* também tornaram essa tecnologia mais acessível aos desenvolvedores e cientistas de dados. Os *frameworks* geralmente têm um número de redes pré-configuradas para aplicativos de exemplo, como reconhecimento de imagem, mas as ferramentas têm potencial para ir além e expandir o aprendizado de máquina para diversas áreas.

Atualmente o trabalho de análise de fluxo de redes de computadores tornou-se maçante devido a quantidade de dados que circulam nas redes, dificultando a detecção manual de ameaças reais de padrões incomuns (MARZ; WARREN, 2015). Sendo assim a automação da análise de redes surge como um bom objeto de estudo para o uso de aprendizado profundo, pois tendo em vista que a tecnologia de aprendizado profundo teve um salto de popularidade na última década que impulsionou a criação de diversos *frameworks* de IA, explorar o potencial desses métodos em atividades onde eles ainda não são tão difundidos tornou-se relevante.

Existem diversos *frameworks* voltados para o ramo do aprendizado de máquina e para a tecnologia de aprendizagem profunda, porém nota-se que há uma tendência ao estímulo do processamento de dados de imagem e áudio por consequência dos bons avanços do método sobre esses tipos de dados (FADLULLAH et al., 2017). Acreditando ser importante uma maior expansão do uso dessas ferramentas em processamento de outras formas dados, este trabalho visa estudar formas de utilizar um conjunto de ferramentas de aprendizagem profunda na criação de um modelo para detecção de anomalias em redes de computadores.

Este trabalho está disposto da seguinte forma. No Capítulo 2, foram estudados os conceitos de anomalias em redes, sua relação com a segurança das redes e os métodos para sua detecção, foram relatados conceitos de aprendizado de máquina e alguns de seus algoritmos, além de uma descrição de aprendizado profundo e das redes neurais, e por fim é feita uma análise de trabalhos relacionados. No Capítulo 3, toda a metodologia baseada no fluxo de trabalho de aprendizado de máquina para redes empregada na criação dos modelos deste trabalho é explicada passo a passo, é feita também uma descrição da base de dados utilizada nos treinamentos bem como os parâmetros utilizados no tratamento dos dados e na seleção de variáveis para o treinamento dos modelos. No Capítulo 4, são demonstrados os resultados do testes aplicados aos modelos treinados, juntamente com uma análise feita sobre os resultados obtidos. Por fim, o Capítulo 5, contém a discussão e considerações finais a cerca do que foi desenvolvido no trabalho.

# 2 APRENDIZADO DE MÁQUINA PARA DETECÇÃO DE ANOMALIAS EM REDES

Este capítulo descreve as noções básicas sobre anomalias em redes bem como estratégias utilizadas para a sua detecção. São apresentadas definições, causas e tipos de anomalias, e os principais algoritmos de aprendizado de máquina aplicados na classificação do tráfego de rede e categorias de aprendizado. Uma descrição sobre DL, o principal método utilizado neste trabalho, também será apresentada. Por fim, é apresentado um conjunto de soluções para classificação de tráfego de rede e detecção de anomalias propostos em trabalhos relacionados.

## 2.1 ANOMALIAS EM REDES

As redes de computadores possuem características em seu fluxo de dados que geram padronização de seu tráfego, características essas que são geradas através de estatísticas retiradas e calculadas através do monitoramento da rede em um determinado intervalo de tempo, algo que pode ser visto como um perfil da rede (SHON; MOON, 2007). Anomalias em redes são marcadas por desvios repentinos e acentuados causados no tráfego em decorrência de diversas situações como falhas em equipamentos, uso excessivo de recursos da rede, problemas com software, hardware e ataques maliciosos, que tem como consequência falta de segurança, problemas de confiabilidade e instabilidade da rede (CHANDOLA; BANERJEE; KUMAR, 2009). Este conjunto de situações anômalas é difícil detecção, porém a identificação de anomalias pode ser feita através da identificação de alterações no padrão do tráfego da rede (BHUYAN; BHATTACHARYYA; KALITA, 2013).

### 2.1.1 Principais Causas de Anomalias

As anomalias podem apresentar causas e consequências variadas, o que dificulta a identificação e solução dos problemas provocados por elas. Segundo (THOTTAN; JI, 2003) as anomalias podem ser causadas por diferentes problemas, que levam à classificação delas em duas categorias. O primeiro grupo reúne as anomalias onde não há a presença de agentes maliciosos. O segundo traz as anomalias causadas por ataques, normalmente arquitetados por agentes que visam romper as barreiras de segurança da rede.

Neste trabalho o interesse está voltado para detecção do segundo grupo de anomalias, onde temos como exemplos os seguintes casos:

- Ataque de força bruta: ataque onde força-se a entrada em algum sistema, site, ser-

vidor, aplicativo, etc. A técnica se dá através de sucessivas tentativas de acertar uma combinação de senha (uma chave), e assim conseguir acesso às informações e dados que deseja. Essa quantidade excessiva de tentativas de acessos gera uma anomalia na rede.

- Ataque de negação de serviço (*Denial of Service*, DoS): diferentemente da maioria dos ataques da Internet, um ataque de negação de serviço não visa invadir um computador para extrair informações confidenciais nem para modificar o conteúdo armazenado neste computador. Tais ataques têm como objetivo tornar inacessíveis os serviços providos pela vítima a usuários legítimos. Um exemplo de ataque DoS é o ataque de negação de serviço distribuído (*Distributed Denial of Service*, DDoS), ocorre quando diversas estações ou máquinas ligadas a rede agem com um objetivo comum de atacar uma determinada vítima para esgotar seus recursos e tornar seus serviços inacessíveis.

- Varredura de portas: consiste no envio de solicitações de clientes para vários endereços de porta de servidor em um *host*, com o objetivo de encontrar uma porta. A ideia é investigar o maior número possível de portas e acompanhar aquelas que são receptivas ou úteis para sua necessidade específica. Está técnica apesar de ser utilizada por administradores de rede para executar tarefas de gerência, pode também ser utilizada por indivíduos maliciosos para encontrar uma porta ativa e explorar uma vulnerabilidade conhecida desse serviço.

### 2.1.2 Anomalias e Segurança

A detecção de anomalias tem amplas aplicações em áreas como como detecção de fraude para cartões de crédito, detecção de intrusão para cibersegurança e vigilância militar para atividades inimigas (BHUYAN; BHATTACHARYYA; KALITA, 2013). A segurança das redes de computadores é importância para preservação dos dados, tanto de instituições como de pessoas, pois as redes se tornaram o principal meio de transporte de informações e, em meio a essa massa de dados que trafega pelas redes, sem dúvida há informações de valor que acabam atraindo o interesse de agentes mal intencionados. Quando estes agentes conseguem acesso indevido em uma rede, podem fazer uso de vulnerabilidades da rede para prejudicar os usuários direta ou indiretamente.

A segurança é um processo complexo, com componentes tecnológicos e humanos, que envolvem estudo de metodologias e comportamentos. A confiabilidade de uma rede está diretamente ligada a capacidade dela de lidar com a prevenção, detecção e recuperação de anomalias. Veja a seguir uma descrição destas capacidades.

- Prevenção
  - Proteção de hardware: comumente chamada de segurança física, nega acessos físicos não autorizados à infraestrutura da rede, tendo como objetivo prevenir roubos de dados, desligamento de equipamentos e demais danos que podem vir a serem efetuados fisicamente no local;
  - Proteção de arquivos e dados: proporcionada pelos serviços de autenticação, controle de acesso e sistemas antivírus. No processo de autenticação, é efetuada a verificação da identidade do usuário; o controle de acesso disponibiliza apenas os serviços pertinentes ao usuário e os programas antivírus garantem a proteção do sistema contra programas maliciosos;
  - Proteção de perímetro: ferramentas de *firewall* e *routers* cuidam desse aspecto, mantendo a rede protegida contra tentativas de intrusão (interna e externa à rede).
- Detecção
  - Alertas: sistemas de detecção de intrusões (IDS) alertam os responsáveis pela segurança sobre qualquer sinal de invasão ou alteração suspeita no perfil da rede que possa indicar um padrão de ataque. Os avisos podem ser emitidos via e-mail, mensagem no console de gerência, celular, etc.;
  - Auditoria: periodicamente deve-se analisar os componentes críticos do sistema a procura de mudanças suspeitas. Esse processo pode ser realizado por ferramentas que procuram, por exemplo, modificações no tamanho dos arquivos de senhas, usuários inativos, etc.
- Recuperação
  - Cópia de segurança dos dados (*Backup*): manter sempre atualizados e testados os arquivos de segurança em mídia confiável e separados física e logicamente dos servidores;
  - Aplicativos de *Backup*: ferramentas que proporcionam a recuperação rápida e confiável dos dados atualizados em caso da perda das informações originais do sistema;
  - *Backup* do Hardware: a existência de hardware reserva, fornecimento autônomo de energia, linhas de dados redundantes, etc., podem ser justificados levando-se em conta o custo da indisponibilidade dos sistemas.

### 2.1.3 Sistemas de Detecção de Intrusão

Os IDS são um importante objeto em nosso estudo, pois são meios técnicos de implementar métodos para análise de rede com intuito de identificar acessos desautorizados que podem ter por finalidade a execução de ataques. É importante o desenvolvimento de automação dos processos de identificação de anomalias, pois, segundo (RYZA et al., 2017), embora seja bastante óbvio quando um computador está sendo bombardeado com o tráfego, detectar uma intrusão pode ser algo significativamente difícil. Analisar a grande quantidade de dados que passa por uma rede em tempo hábil para detecção de um ataque é algo inviável para um ser humano. É com intuito de tornar viável a detecção de anomalias que há décadas cientistas da tecnologia desenvolvem ferramentas de detecção de intrusão baseadas em diversos métodos e algoritmos.

Os sistemas de detecção de intrusão, segundo (ALVES et al., 2016) podem ser divididos em dois grupos:

1. Detecção de intrusão baseada em assinatura - Esses sistemas comparam o tráfego recebido com um banco de dados preexistente de padrões de ataques conhecidos como assinaturas, semelhante ao software antivírus.

2. Detecção de intrusão com base em anomalias - Usa estatísticas formadas através do fluxo da rede em determinados intervalos de tempo para traçar perfis da rede, e então detectar anomalias no fluxo através de alterações nos perfis. Este sistema usa o aprendizado de máquina para criar um modelo que simula atividades regulares e, em seguida, compara o novo comportamento com o modelo existente.

Os IDS detectam invasões em lugares diferentes e podem ser classificados com base em onde eles operam. Detecção de intrusão de rede (NIDS), um sistema estrategicamente colocado (em um ou múltiplos locais) para monitorar todo o tráfego da rede. Detecção de intrusão de *host* (HIDS) executado em todos os dispositivos da rede conectados à Internet-intranet da organização, podem detectar tráfego malicioso originado de dentro (por exemplo, quando um *malware* está tentando se espalhar para outros sistemas a partir de um *host* em uma organização).

Há também uma classificação dos IDSs com base em suas ações. Ativo, também conhecido como sistema de detecção e prevenção de intrusões, gera alertas, registra entradas e executa comandos para alterar configurações em tentativa de proteger a rede. Passivo, apenas detecta atividade maliciosa e gera um alerta ou logs, mas não executa nenhuma ação.

## 2.2 APRENDIZADO DE MÁQUINA

Aprendizagem de máquina (*Machine Learning*, ML) é um subconjunto da inteligência artificial, onde o sistema é configurado para aprender e pensar como um ser humano. A informação inicial é dada ao sistema, onde o algoritmo pode aprender os dados e sua classificação, o objetivo final do sistema é fazer as suas próprias decisões no futuro (semelhante a um ser humano). ML trabalha com certo grau de probabilidade, com base nos dados que são analisados e as decisões que o sistema irá adotar, seu núcleo está treinado para fazer previsões e sua capacidade é de prever eventos futuros com base em eventos passados (NGUYEN; ARMITAGE, 2008).

Lidar com problemas complexos é uma das vantagens mais importantes de aprendizado de máquina. Para algumas tarefas que requerem classificação, regressão e tomada de decisão, aprendizagem de máquina pode gerar algoritmos que alcançam resultados próximos ou até melhores do que os seres humanos em determinadas tarefas (WANG et al., 2017). A ML permite o desenvolvimento de sistemas que são significativamente difíceis e caros para serem construídos manualmente por necessitarem de habilidades ou conhecimentos detalhados, ajustados especificamente para uma determinada tarefa. Segundo consta em (HAQ et al., 2015) existem três categorias de ML, são elas:

- Aprendizado supervisionado: aplicável quando são conhecidos pares associando entradas às respectivas respostas desejadas. Algoritmos nesta categoria buscam aprender uma função que mapeia as entradas nas saídas desejadas, e uma vez aprendida, tal função de mapeamento pode ser aplicada a novas entradas. O processo de treinamento pode ser aprimorado e refeito até que o modelo atinja um nível de precisão desejado nos dados de teste.

- Aprendizado não-supervisionado: aplicável quando não são fornecidas saídas desejadas, mas se deseja encontrar ou fazer inferências sobre padrões inerentes ao comportamento das amostras de entrada. Os algoritmos mais conhecidos nesta categoria são os de clusterização, responsáveis pela organização das amostras de entrada em um conjunto de grupos, ditos *clusters*.

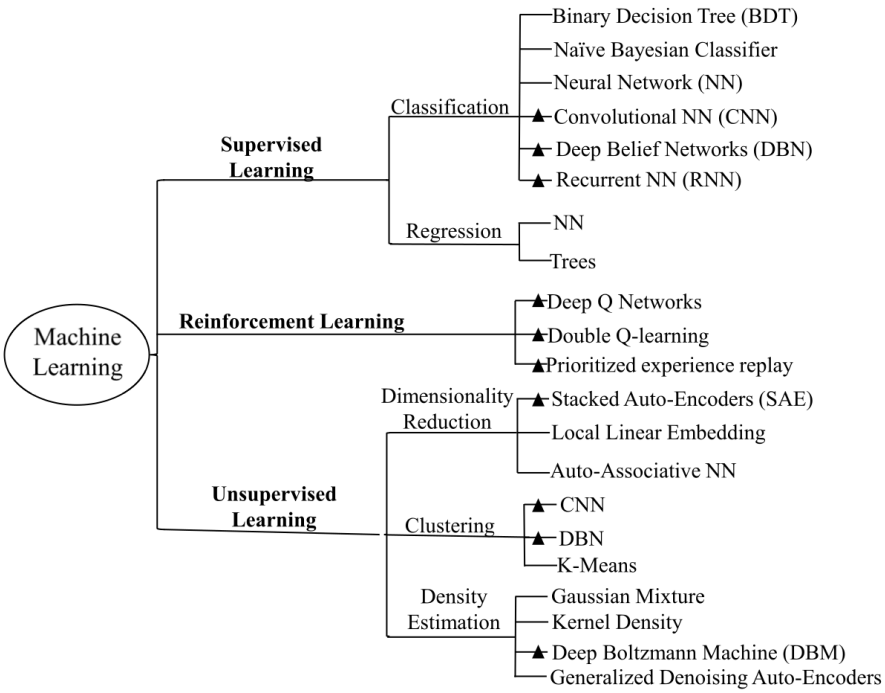
- Aprendizado por reforço: aplicável quando se deseja um algoritmo que interage com um ambiente para tomar decisões. Nesta categoria o aprendizado se dá expondo o algoritmo a um ambiente no qual ele pode praticar por tentativa e erro, recebendo recompensa ou punições como feedback pelas decisões tomadas. Com o passar do tempo, passa a acumular conhecimento com a experiência passada, ou seja, aprende a preferir o tipo certo de ação e evitar as que induzem ao erro, tentando capturar o melhor conhecimento/estratégia para tomar decisões futuras.

- Aprendizado semi-supervisionado: segundo consta em (CHAPELLE; SCHOLKOPF; ZIEN, 2009) e (ABNEY, 2007), há mais uma categoria de ML, o aprendizado semi-

supervisionado, utilizado para as mesmas aplicações que o aprendizado supervisionado, mas este aqui manipula tanto dados rotulados quanto não-rotulados. Normalmente faz uso de uma pequena quantidade de dados rotulados com uma grande quantidade de dados não-rotulados, o que é útil quando o custo associado à rotulação é elevado para possibilitar um processo de treinamento totalmente rotulado.

A Figura 3.3 demonstra de forma simplificada os caminhos oferecidos pelos métodos de aprendizado de máquina.

Figura 2.1 – Organização dos tipos de aprendizado de máquina.




Fonte: Retirado de (FADLULLAH et al., 2017).

Desde o início do ML, pesquisadores vem trabalhando em experimentos e desenvolvendo soluções para as mais diversas tarefas em várias áreas da computação, como sistemas de recomendação, reconhecimento de manuscrito, processamento de linguagem natural, dentre outros. Na segurança de TI, o ML também é amplamente utilizado, muitas das soluções mais avançadas que existem hoje, como ferramentas de análise de comportamento de usuário, usam algoritmos de ML para identificar potenciais ataques. O ML estabelece uma linha de base do comportamento normal que usa para detectar anomalias, permitindo que potencialmente as organizações identifiquem e atuem sobre ameaças. Segundo estudo feito por (ASHRAF; AHMAD; ASHRAF, 2018) os algoritmos de ML com maior número de experimentos em detecção de anomalias são:

- *Support Vector Machines* (SVM, Máquina de Vetores de Suporte): tem como objetivo encontrar a melhor função de classificação entre membros de duas classes utilizando

os dados de treino. O SVM calcula o hiperplano que define uma fronteira entre as classes, ele maximiza a fronteira que separa os dados das duas classes e a classificação ocorre ao analisar se os dados estão acima ou abaixo deste hiperplano.

- *Decision Tree* (DT, Árvore de Decisão): classifica as instâncias ao desenhar os diferentes atributos abaixo da raiz. Cada ramo representa uma escolha, valor possível para este atributo; Enquanto isso cada folha representa uma decisão. O processo de inferência começa em sua raiz, e prossegue para uma folha. Cada atributo é designado a um nó, e na folha há um resultado possível (estado provável). Ele é usado em aprendizado supervisionado, tanto para classificação, quanto para regressão.

- *Naive Bayes*: é um algoritmo de classificação probabilístico que parte do princípio de que as características são independentes, significando que a ocorrência de uma característica não afeta a probabilidade de ocorrência de outra. O modelo possui uma tabela de probabilidades para cada característica, definidas a partir da base de treinamento. É possível determinar a probabilidade ao se analisar o conjunto de dados de treino utilizando o teorema de Bayes.

- Redes neurais artificiais (RNA): também chamadas de sistemas conexionistas, são sistemas de computação inspirados, a redes neurais biológicas que constituem cérebros de animais. Esses sistemas ”aprendem” a executar tarefas considerando exemplos, geralmente sem serem programados com regras específicas da tarefa. O conceito das RNA deu base para o desenvolvimento do DL.

### 2.2.1 Aprendizado Profundo

Um subgrupo específico de técnicas de ML são chamadas de DL, geralmente utilizam redes neurais (RN) profundas e dependem de uma quantidade significativa de dados para o treinamento em relação a maioria dos algoritmos de ML. Existem certos fatores que diferenciam as técnicas clássicas de ML das técnicas de DL, segundo (KARATAS; DEMIR; SAHINGOZ, 2018). A principal diferença entre aprendizado profundo e aprendizado de máquina decorre da maneira como os dados são apresentados ao sistema. Os algoritmos de aprendizado de máquina quase sempre exigem dados estruturados, enquanto as redes de aprendizado profundo dependem de camadas da RNA (redes neurais artificiais).

Os algoritmos de aprendizado de máquina são criados para "aprender"a fazer as coisas, compreendendo os dados rotulados e , em seguida, usá-los para produzir resultados adicionais com mais conjuntos de dados. No entanto, eles precisam ser reciclados por meio de intervenção humana quando o resultado real não é o desejado. As redes de aprendizado profundo não requerem intervenção humana, pois as camadas aninhadas

nas redes neurais colocam os dados em hierarquias de conceitos diferentes, que eventualmente aprendem com seus próprios erros. No entanto, ainda estão sujeitos a resultados defeituosos se a qualidade dos dados não for boa o suficiente.

Em ambos os casos é a qualidade dos dados que determina a qualidade do resultado, pois esses dois subconjuntos de IA giram em torno de dados para realmente fornecer qualquer forma de "inteligência". No entanto, o que se deve saber é que o aprendizado profundo exige muito mais dados do que um algoritmo tradicional de aprendizado de máquina. A razão para isso é que ele só é capaz de identificar arestas (conceitos, diferenças) em camadas de redes neurais quando expostas a uma quantidade consideravelmente maior em relação a um algoritmo de aprendizado de máquina.

Segundo (SCHMIDHUBER, 2015) DL possui origens datadas da década de 1960, onde pesquisadores começaram a modelar redes profundas baseadas em descobertas da época sobre o córtex visual de gatos. Apesar de estudos relacionados as redes profundas existirem à muitas décadas, a grande quantidade de dados necessária para o treinamento de uma rede neural profunda, juntamente com o poder computacional exigido para o processamento das redes fizeram com que elas acabassem em segundo plano por um longo tempo. Ao que consta em (Data Science Academy, 2019) no início dos anos 2000, o poder computacional expandiu-se exponencialmente e a explosão de novas técnicas computacionais que surgiram no mercado possibilitou a emergência do DL.

Atualmente o DL está sendo relevante em diversas áreas de estudos, tendo nas duas últimas décadas a inclusão de diversas ferramentas computacionais que facilitam a criação de modelos de DL. Segundo (VASCONSELOS, 2017) existem três fatores que convergiram para que o DL se tornasse viável: a grande quantidade de dados disponíveis (Big Data); o desenvolvimento de novas técnicas de DL; e a supercomputação de forma acessível. Neste último quesito, as GPUs tem total importância, pois são as protagonistas nesta revolução: além de serem processadores altamente paralelos, suas arquiteturas permitem que diversas tarefas típicas do DL sejam beneficiadas de forma direta pelo paralelismo disponível.

### 2.2.2 Redes Neurais Profundas

As RN profundas são inspiradas nas RN simples e o conceito de neurônio é o mesmo nos dois casos. Um neurônio é uma unidade de processamento de informação fundamental para operação de uma RN. Os três elementos básicos do modelo neural são identificados em (HAYKIN, 2007) de tal forma:

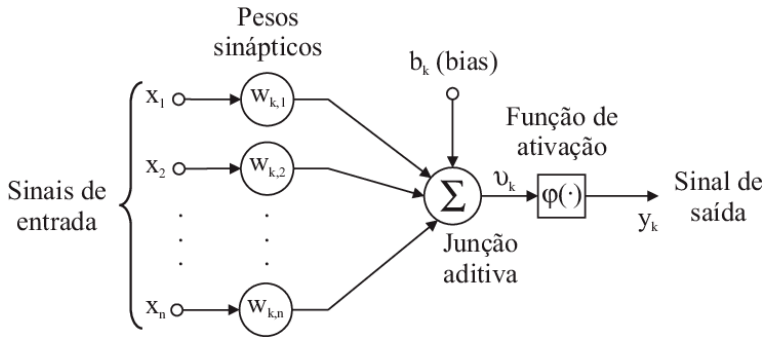
1. Um conjunto de sinapses ou elos de conexão, cada uma caracterizada por um peso ou força própria. Especificamente, um sinal $x$ na entrada da sinapse conectada ao neurônio é multiplicado pelo peso sináptico **w**. Ao contrário de uma sinapse do

cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos ou positivos.

2. Um somador para somar os sinais de entrada, ponderadas pelas respectivas sinapses do neurônio.

3. Uma função de ativação para restringir a amplitude da saída de um neurônio. A função de ativação é também referida como função restritiva, pois restringe o intervalo permissível de amplitude do sinal de saída a um valor finito. Tipicamente, o intervalo normalizado da amplitude de saída de um neurônio é escrito como intervalo unitário fechado [0,1] ou alternativamente [-1,1].

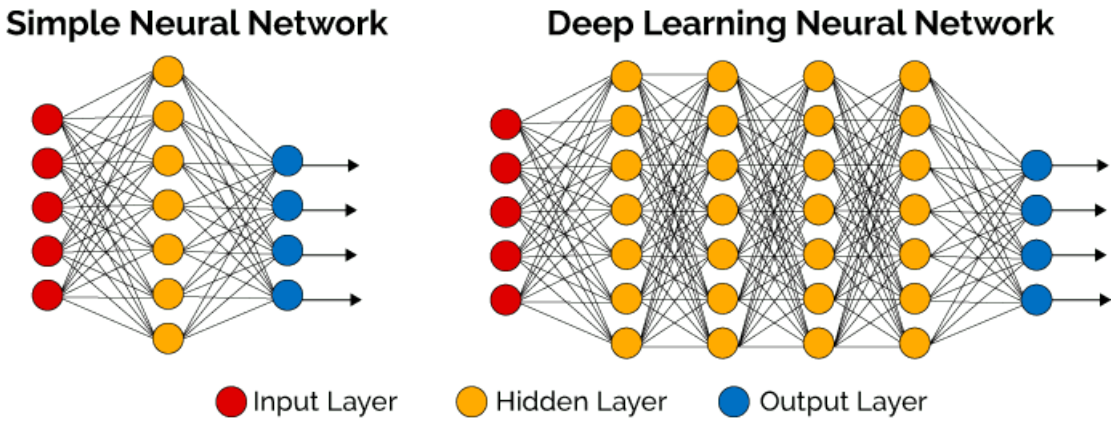
Figura 2.2 – Neurônio artificial com sinais de entrada x e pesos sinápticos w.




Fonte: Retirado de Zanetti et al. (2012).

O modelo neural inclui também um bias que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo. Podemos, portanto, reformular o modelo do neurônio. O efeito do bias é levado em conta de duas maneiras: (i) adicionando-se um novo sinal de entrada fixo em +1 e (ii) adicionando-se um novo peso sináptico igual ao bias (HAYKIN, 2007).

Figura 2.3 – Diferença entre RN simples e RN profunda.




Fonte: Adaptado de Data Science Academy (2019).

Como mostra a Figura 2.2 as RN simples são compostas de três camadas de neurônios: camada de entrada, camada oculta e camada de saída. Nas RN profundas a diferença é o número maior de camadas intermediárias.

## 2.3 TRABALHOS RELACIONADOS

Em (HAAS et al., 2019) foi proposto o desenvolvimento de um sistema baseado na arquitetura Lambda, capaz de processar e analisar o fluxo de dados do tráfego de rede, com o propósito de identificar anomalias de rede causadas por ataques de DDoS, força bruta e varredura de portas, sobre determinados protocolos de rede. O sistema realiza a integração de diferentes ferramentas de código aberto onde cada ferramenta é responsável por determina funcionalidade, esse sistema utiliza o algoritmo K-means na classificação dos dados enquanto que no presente trabalho é utilizado o algoritmo perceptron multicamadas, porém os dois algoritmos são utilizados com o mesmo propósito, detectar anomalias.

O algoritmo DT foi utilizado em (VILLANO, 2018) para avaliar o quanto ele é capaz de ajudar na previsão e identificação de ataques de rede. O objetivo principal do experimento foi avaliar o desempenho do algoritmo em três diferentes cenários que contam com quantidades diferentes de atributos de entrada para treinamento, que foram retirados de um conjunto de dados, no presente trabalho também são analisados três diferentes cenários, porém eles diferem-se do trabalho relacionado, pois são diferentes entre si na quantidade de registros e de classes utilizados no treinamento dos classificadores. O trabalho relacionado alcançou uma precisão de 98,2% durante a fase de treinamento e 86,1% em uma base de dados de teste, nesse modelo 9 atributos em 19 foram usados para criar a árvore.

A metodologia empregada no presente trabalho tem como base o passo a passo descrito em (WANG et al., 2017), onde é fornecido um resumo do fluxo de trabalho básico para criação de modelos de ML para redes, uma pesquisa seletiva das representações mais avançadas do ML empregado em redes e uma descrição de como eles executam cada etapa no fluxo de trabalho. Um trabalho de pesquisa sobre literatura focada no levantamento dos métodos de ML e mineração de dados para suporte na detecção de intrusão é feito em (BUCZAK; GUVEN, 2015), baseado em número de citações ou relevância de um método emergente, ele aborda temas como conjuntos de dados e complexidade de algoritmos, essa pesquisa serviu como base de consulta para o presente trabalho. O documento de pesquisa (NGUYEN; ARMITAGE, 2008) também analisa pesquisas emergentes sobre ML na classificação de tráfego de rede. O documento traz a revisão de 18 trabalhos significativos em um período de 2004 a início de 2007 que estão listados da Figura 2.4.

Figura 2.4 – Tabela de trabalhos analisados em (NGUYEN; ARMITAGE, 2008).

| ML/DM Method | Paper | No. of Times Cited (as of 7/28/2015) | Cyber Approach | Data Used |
|---|---|---|---|---|
| ANN | Cannady [24] | 463 | misuse | Network packet-level data |
| ANN | Lippmann & Cunningham [27] | 235 | anomaly | DARPA 1998 |
| ANN | Bivens et al. [28] | 135 | anomaly | DARPA 1999 |
| Association rules | Brahmi et al. [32] | 3 | misuse | DARPA 1998 |
| Association rules | Zhengbing et al. [33] | 33 | misuse | Attack signatures (10 to 0 MB) |
| Association rules | Apiletti et al. [35] | 14 | anomaly | NetFlow |
| Association rules – Fuzzy | Luo and Bridges [39] | 192 | anomaly | tcpdump |
| Association rules – Fuzzy | Tajbakhsh et al. [38] | 124 | hybrid | KDD 1999 (corrected) |
| Bayesian network | Livadas et al. [42] | 208 | misuse | tcpdump – botnet traffic |
| Bayesian network | Jemili et al. [43] | 31 | misuse | KDD 1999 |
| Bayesian network | Kruegel et al. [44] | 260 | anomaly | DARPA 1999 |
| Clustering – density based | Hendry and Yang [50] | 6 | misuse | KDD 1999 |
| Clustering – density based | Blowers and Williams [51] | 2 | anomaly | KDD 1999 |
| Clustering – sequence | Sequeira and Zaki [52] | 214 | anomaly | shell commands |
| Decision tree | Kruegel and Toth [55] | 155 | misuse | DARPA 1999 |
| Decision tree | Bilge et al. [56] | 187 | anomaly | DNS data |
| Ensemble learning | Mukkamala et al. [65] | 255 | misuse | DARPA 1998 |
| Ensemble – Random Forest | Gharibian and Ghorbani [63] | 19 | misuse | KDD 1999 |
| Ensemble – Random Forest | Bilge et al.[66] | 49 | anomaly | NetFlow |
| Ensemble – Random Forest | Zhang et al. [62] | 92 | hybrid | KDD 1999 |
| Evolutionary Comp - GA | Li [73] | 235 | misuse | DARPA 2000 |
| Evolutionary Comp - GP | Abraham et al. [74] | 83 | misuse | DARPA 1998 |
| Evolutionary Comp - GP | Hansen et al. [75] | 52 | misuse | KDD 1999 – subset |
| Evolutionary Comp - GA | Khan [76] | 15 | anomaly | KDD 1999 |
| Evolutionary Comp - GP | Lu and Traore [78] | 124 | anomaly | DARPA 1999 |
| HMM | Ariu et al. [82] | 25 | misuse | HTTP payload |
| HMM | Joshi and Phoha [83] | 61 | anomaly | KDD 1999 |
| Inductive learning | Lee et al. [87] | 1358 | misuse | DARPA 1998 |
| Inductive learning | Fan et al. [88] | 195 | anomaly | DARPA 1998 |
| Naïve Bayes | Benferhat et al. [45] | 17 | anomaly | DARPA 2000 |
| Naïve Bayes | Panda and Patra [90] | 90 | misuse | KDD 1999 |
| Naïve Bayes | Amor et al. [91] | 277 | anomaly | KDD 1999 |
| Sequence mining | Hu and Panda [93] | 151 | misuse | Database logs |
| Sequence mining | Li et al. [94] | 18 | anomaly | DARPA 2000 |
| SVM | Li et al. [96] | 56 | misuse | KDD 1999 |
| SVM | Amiri et al. [97] | 84 | misuse | KDD 1999 |
| SVM – Robust | Hu et al. [98] | 114 | anomaly | DARPA 1998 |
| SVM | Wagner et al. [99] | 1 | anomaly | NetFlow |
| One-class SVM and GA | Shon and Moon [101] | 166 | hybrid | DARPA 1999 |

Fonte: Adaptado de (NGUYEN; ARMITAGE, 2008).

Com relação à aprendizagem profunda, (FADLULLAH et al., 2017) examina um conjunto de trabalhos sobre aprendizagem profunda aplicada em tráfego de rede, fornecendo uma visão geral dos conhecimentos aprofundados e algoritmos relevantes para o tráfego de rede. Esse trabalho serviu como base para captação de informações para o presente trabalho em se tratando de DL com aplicação em redes.

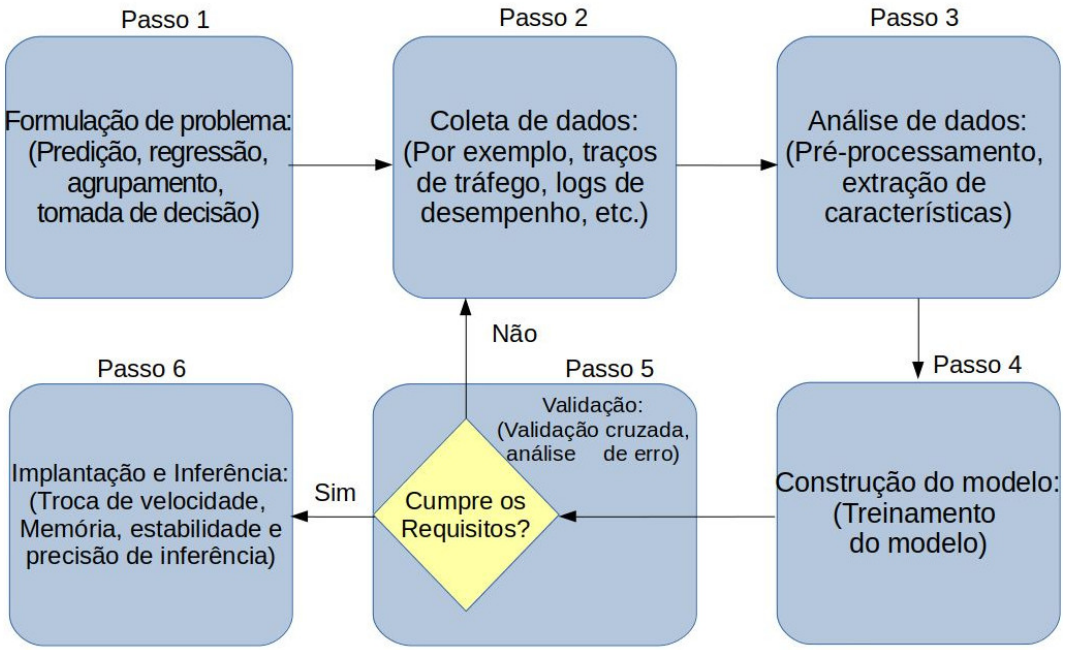
# 3 METODOLOGIA

Este capítulo descreve todo o processo de desenvolvimento do modelo de treinamento da RN profunda, com uma descrição do fluxo de trabalho típico de aprendizado de máquina para rede (*machine learning for networking*, MLN), definição do problema, escolha do algoritmo, obtenção de dados, tratamento de dados, treinamento e teste do algoritmo. É também realizada uma introdução à base de dados utilizada, bem como uma explicação acerca das métricas de processamento dos dados de entrada e ferramentas de software empregadas no processo de manipulação dos dados. Por fim este capítulo aborda na seção 3.2 Avaliação, o modo como foram manipulados os dados e os classificadores com intuito de gerar dados para análise.

## 3.1 O FLUXO DE TRABALHO DE ML PARA REDE

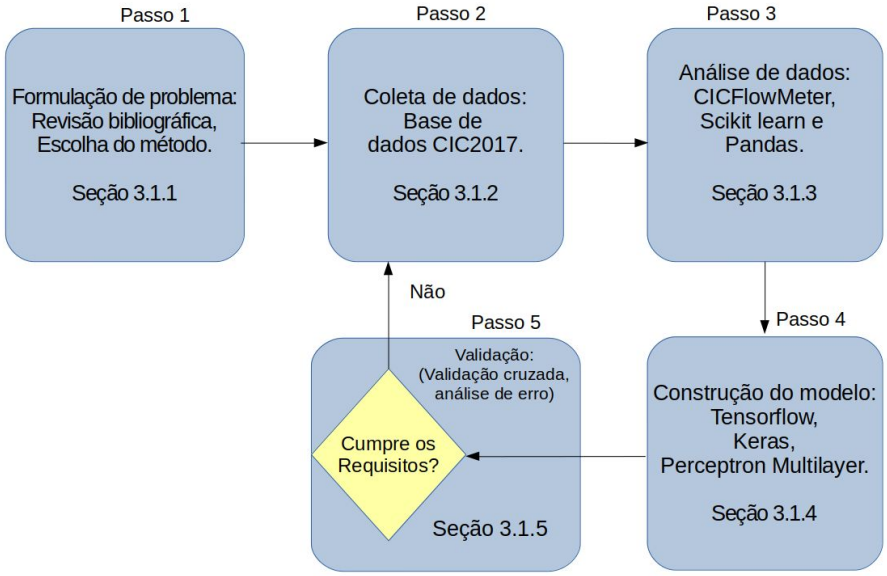
O fluxo de trabalho típico do ML empregado em rede é descrito em (WANG et al., 2017) em seis etapas. Nesse trabalho serão executadas as cinco primeiras etapas, sendo assim não será executada a etapa de implantação e inferência. Com base nesta descrição será feita uma explicação acerca de cada etapa deste fluxo e como cada etapa foi executada no presente trabalho.

Figura 3.1 – Diagrama dos passos do MLN.




Fonte: Adaptado de (WANG et al., 2017).

Figura 3.2 – Diagrama dos passos do MLN deste trabalho.




Fonte: Adaptado de (WANG et al., 2017).

### 3.1.1 Formulação de Problema

O processo de aprendizagem de máquina consome tempo relevante e envolve alto custo, é importante abstrair e formular o problema na primeira etapa da MLN. Nessa etapa é relevante identificar que tipo de problema tem-se e que categoria de aprendizado (por exemplo classificação, agrupamento ou tomada de decisão) poderá fornecer os melhores resultados para o modelo.

Neste trabalho é feito uso de métodos para classificação binária e classificação multi-classe, onde para o método binário estima-se classificar os registros em benignos ou malignos, e para o método multi-classe estima-se classificar os registros entre as quinze classes disponíveis na base de dados que está descrita na seção 3.2 Avaliação. Os dois métodos enquadram-se no aprendizado supervisionado, pois o treinamento é executado sobre dados pré rotulados, para esse trabalho foi utilizado o algoritmo perceptron multicamadas, pois a classificação é um caso particular de regressão quando a variável resposta é categórica, e perceptron multicamadas são bons algoritmos classificadores (FADLULLAH et al., 2017).

### 3.1.2 Coleta de Dados

O objetivo desta etapa é coletar dados representativos da rede, eles podem ser capturados a partir de diferentes camadas da rede de acordo com as necessidades da aplicação. Por exemplo, o problema de classificação de tráfego geralmente requer conjuntos de dados contendo rastreamentos em nível de pacote rotulados com as classes de aplicação correspondentes. No contexto da MLN, os dados são frequentemente coletados em duas fases. Na fase *offline*, coletar uma quantidade suficientemente grande de dados históricos de alta qualidade é importante para a análise e treinamento de modelos. Na fase *online*, informações de desempenho e estado da rede em tempo real, são frequentemente usados como entradas ou sinais de *feedback* para o modelo de aprendizagem. Os dados recém-coletados também podem ser armazenados para atualizar o conjunto de dados históricos para adaptação do modelo.

Com relação coleta de dados neste trabalho, foi feito uso do conjunto de dados denominado CICIDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018) desenvolvido por *Canadian Institute for Cybersecurity*. Este conjunto é relativamente atualizado e contém ataques comuns e fluxos benignos, que se assemelham aos dados do mundo real, segundo consta em sua documentação gerar tráfego realista em segundo plano foi a principal prioridade na criação desse conjunto de dados. Para esse conjunto de dados, foi simulado o comportamento abstrato de 25 usuários com base nos protocolos HTTP (*Hypertext Transfer Protocol*), HTTPS (*Hyper Text Transfer Protocol Secure*), FTP (*File Transfer Protocol*), SSH (*Secure Shell*) e e-mail.

### 3.1.3 Análise de Dados

Parte do processo é analisar os dados e selecionar os atributos ,também chamados variáveis, relevantes para a solução do problema, esses atributos são responsáveis por gerar um modelo de conhecimento consistente e generalista. Segundo (GUYON; ELISSEEFF, 2003) existem muitos benefícios potenciais na seleção de atributos, por exemplo a simplificação da visualização e compreensão dos dados, redução dos requisitos de medição e armazenamento, redução dos tempos de treinamento e utilização, para melhorar o desempenho da previsão. É muito comum nesse momento, excluir certos dados para não causar confusão nem *overfitting* que ocorre quando consegue-se um bom ajuste do modelo nos dados de treinamento, mas ele não generaliza bem em dados novos, dados provenientes de uma fonte diferente da base utilizada no treinamento.

Existem vários métodos que podem ser adotados no processo de seleção de variáveis. Em (SHARAFALDIN; LASHKARI; GHORBANI, 2018) são fornecidos resultados de análise do tráfego de rede em CSV (valores separados por vírgulas), com fluxos rotulados com base no carimbo de data / hora, IPs de origem e destino, portas de origem e des-

tino, protocolos e ataques. Essas análises foram geradas através do CICFlowMeter, uma ferramenta que gera mais de oitenta atributos estatísticos em CSV de um fluxo de rede retirados de arquivos PCAP e calculados em lotes delimitados por determinado espaço de tempo, este tempo pode ser configurado junto a ferramenta (DRAPER-GIL et al., 2016).

Segundo (SHARAFALDIN; LASHKARI; GHORBANI, 2018), a classe RandomForestRegressor do scikit-learn (PEDREGOSA et al., 2011; BUITINCK et al., 2013) pode ser utilizada para seleção dos melhores atributos, a utilização deste método é proposta em um cenário onde primeiro calcula-se a importância de cada recurso em todo o conjunto de dados, então o resultado final é alcançado multiplicando a padronização (Normalização do escore Z) de cada recurso para cada classe, com a importância do valor do recurso correspondente. A normalização do escore Z é calculada pelo valor do recurso x, menos a média m dos valores do recurso, dividido pelo desvio padrão (sigma) do recurso. Após a aplicação desse processo foram retirados os quatro atributos mais relevantes para cada classe, o que após a exclusão das duplicatas resultou em vinte e três atributos que são descritos no capítulo 4 Avaliação.

$$Z = \frac{x - m}{\sigma} \tag{3.1}$$

### 3.1.4 Construção do Modelo

A construção do modelo envolve a seleção de algoritmos, formação e ajuste, e o modelo deve ser selecionado de acordo com a categoria do problema e quantidade de dados disponíveis. O processo de seleção e algoritmo adotado já foram descritos na subseção 3.1.1 referente a formação do problema. O processo de ajuste é realizado sobre hiper-parâmetros do algoritmo com a finalidade de melhorar o desempenho na resolução do problema. Existem certos ajustes possíveis em algoritmos de ML como taxa de aprendizado, tamanho do lote, otimizadores e decaimento de pesos sinápticos, os hiperparâmetros agem como botões que podem ser ajustados durante o treinamento do modelo, e para que o modelo forneça o melhor resultado, é preciso encontrar o valor ideal desses hiper-parâmetros (PONTI; COSTA, 2018). Uma breve descrição dos hiper-parâmetros modificados durante a avaliação das melhores configurações dos modelos utilizados neste estudo.

- Tamanho do lote (*Batch Size*): é um termo usado em aprendizado de máquina e refere-se ao número de exemplos de treinamento usados em uma iteração (Data Science Academy, 2019).

- Época: corresponde a um intervalo de iterações em que todos os elementos do conjunto de entrada tenham passado uma vez pela rede.
- Funções de ativação: introduzem um componente não linear nas redes neurais, que faz com que elas possam aprender mais do que relações lineares entre as variáveis dependentes e independentes.
- Otimizadores: são métodos de atualização de pesos.
- Função de perda: é um método de avaliar quão bem o algoritmo específico modela os dados fornecidos. Se as previsões se desviarem demais dos resultados reais, a função de perda indicará um número grande. Gradualmente, com a ajuda de alguma função de otimização, a função de perda aprende a reduzir o erro na previsão.

A construção de um modelo não é um processo exato, pois são necessários vários testes com configurações diferentes para o desenvolvimento de um modelo satisfatório. Por exemplo alteração do número de neurônios de cada camada, além de valores diferentes para os hiperparâmetros e tempo disponível para treinamento (YOUNG et al., 2015). Nesta fase foram utilizadas as classes KerasClassifier e GridSearchCV, ambas disponibilizadas na biblioteca do Keras, estas classes recebem um conjunto de dados e um conjuntos de hiper-parâmetros como entrada e após testar todas as configurações possíveis com os hiper-parâmetros de entrada retorna a possível melhor configuração para o treinamento da rede.

Quadro 3.1 – Parâmetros testados para classificação multi-classe

| **Funções de ativação** | **Funções de perda** | **Quantidade de Neurônios** | **Tamanho de lote** | **Otimização** | **Épocas** |
|---|---|---|---|---|---|
| relu | categorical crossentropy | 16 | 10 | adam | 10 |
| tanh | | 23 | 100 | adadelta | 20 |

Quadro 3.2 – Parâmetros testados para classificação binária

| **Funções de ativação** | **Funções de perda** | **Quantidade de Neurônios** | **Tamanho de lote** | **Otimização** | **Épocas** |
|---|---|---|---|---|---|
| relu | binary-crossentropy | 12 | 10 | adam | 10 |
| tanh | hinge | 23 | 100 | adadelta | 20 |

Os resultados alcançados na escolha dos hiperparâmetros Quadro 3.3 demonstraram que um tamanho menor de lote gerou um melhor desempenho no treinamento dos classificadores, também é possível notar que uma quantidade de neurônios nas camadas ocultas igual a quantidade de neurônios da camada de entrada convergiu para um melhor resultado no treinamento. Nota-se que o número maior de épocas resultou no melhor desempenho para ambos os modelos.

Quadro 3.3 – Resultados da seleção dos hiper-parâmetros

| | **Binária** | **Multi-classe** |
|---|---|---|
| **Função de ativação** | relu | tanh |
| **Função de perda** | binary-crossentropy | categorical-crossentropy |
| **Quantidade de neurônios** | 23 | 23 |
| **Tamanho de lote** | 10 | 10 |
| **Otimização** | adam | adadelta |
| **Épocas** | 20 | 20 |
| **Precisão alcançada** | 77,42% | 86,54% |

- Otimizações utilizadas:
  - Adadelta: se adapta dinamicamente ao longo do tempo usando apenas informações de primeira ordem e possui sobrecarga computacional mínima, além da descida do gradiente estocástico (ZEILER, 2012).
  - Adam: um algoritmo para otimização de primeira ordem baseada em gradiente de funções objetivas estocásticas, baseadas em estimativas adaptativas de momentos de ordem inferior. O método é simples de implementar, é computacionalmente eficiente, possui poucos requisitos de memória, é invariável ao redimensionamento diagonal dos gradientes, e é adequado para problemas grandes em termos de dados e ou parâmetros (KINGMA; BA, 2014).
- Funções de ativação utilizadas:
  - ReLU: é a unidade linear rectificada, ou seja é uma função linear, é a função de ativação mais amplamente utilizada ao projetar redes neurais atualmente (RAMACHANDRAN; ZOPH; LE, 2017).
  - Tanh: é uma função logística com intervalo de (-1 a 1). A vantagem é que as entradas negativas serão mapeadas fortemente negativas e as entradas zero serão mapeadas perto de zero no gráfico tanh.

### 3.1.5 Validação do Modelo

A validação *offline* é indispensável no fluxo de trabalho da MLN para avaliar se o algoritmo de aprendizagem funciona bem o suficiente. Durante esta etapa, a validação cruzada é geralmente usada para testar a precisão geral do modelo para mostrar se o modelo está sobre-ajustado ou mal adaptado. Os testes aplicados nessa fase fornecem uma boa orientação sobre como otimizar o modelo. Analisar amostras erradas ajuda na compreensão dos erros para determinar se o modelo e os atributos são adequados ou os dados são representativos o suficiente para um problema.

Nesse trabalho foi utilizada a validação cruzada, uma técnica para avaliar modelos de ML por meio de treinamento de vários modelos de ML em subconjuntos de dados de entrada disponíveis e avaliação deles no subconjunto complementar dos dados. O conceito central das técnicas de validação cruzada é o particionamento do conjunto de dados em subconjuntos mutuamente exclusivos, e posteriormente, o uso de alguns destes subconjuntos para a estimação dos parâmetros do modelo (ZHANG, 2011). A classe cross_val_score disponível no Scikit Learn foi utilizada para a fase de validação do modelos, pois possibilita a aplicação automatizada de várias execuções da validação em um conjunto de dados. Neste trabalho foram executadas quatro aplicações de validação cruzada para cada um dos modelos propostos.

## 3.2 AVALIAÇÃO

Para a avaliação deste trabalho três cenários de classificadores foram propostos: classificação binária, onde a base foi dividida em registros malignos e benignos; classificação multi-classe com a aplicação de proporcionalidade aos registros; e classificação multi-classe com exclusão das quatro classes que possuíam menos do que mil registros.

- Cenário 1, classificação binária: foram utilizados um milhão de registros divididos em metade para malignos e metade para benignos.
- Cenário 2, classificação multi-classe com proporcionalidade: foram reescalonadas as classes que possuíam menos ou mais que cem mil registros, deixando todas as classes com a mesma quantidade de registros.
- Cenário 3, classificação multi-classe com exclusão: foram excluídas as classes com menos de mil registros restando onze classes utilizadas com mil registros.

Para ambos os cenário foi criado um modelo de RN com uma camada de entrada, duas camadas ocultas e uma camada de saída, e diante dessa configuração foram feitos testes com hiper-parâmetros diferentes para números de épocas, tamanho de lote, função

de ativação e função de perda, em busca da melhor configuração de hiper-parâmetros para o treinamento dos classificadores. Os atributos usados como entrada no modelo foram os atributos resultantes do processo descrito na subseção 3.1.3, esses atributos estão descritos no Quadro 3.5.

A base de dados está dividida em oito arquivos CSV, esses arquivos foram concatenados e tratados de forma que pudessem ser utilizados como entradas nas redes neurais. Foi realizado um tratamento no desbalanceamento das classes, pois a base de dados contêm uma quantidade desequilibrada de registros por classe como indicado no Quadro 3.4, o que gera um problema onde o modelo tenderá a classificar a maior parte dos registros como pertencentes as classes que possuem quantidade significativamente maior de registros na fase de treinamento (HYUN et al., 2018).

O problema do desbalanceamento é tratado fazendo replicação de dados das classes que possuem menor quantidade de registros ou diminuindo a quantidade de dados das classes que possuem maior quantidade de registros, e posteriormente aplica-se uma ordenação aleatória. No processamento dos dados para treinamento do classificador binário, além do balanceamento das classes, todos os registros que não são considerados como benignos foram rotulados como malignos (YAP et al., 2014).

Quadro 3.4 – Classes rotuladas da base de dados e quantidade de registros.

| **Classe** | **Quantidade** | **Classe** | **Quantidade** |
|---|---|---|---|
| Benign | 2273097 | DoS Slowhttptest | 5499 |
| DoS Hulk | 231073 | Bot | 1966 |
| PortScan | 158930 | Web Attack Brute Force | 1507 |
| DDoS | 128027 | Web Attack XSS | 652 |
| DoS GoldenEye | 10293 | Infiltration | 36 |
| FTP-Patator | 7938 | Web Attack Sql Injection | 21 |
| SSH-Patator | 5897 | Heartbleed | 11 |
| DoS slowloris | 5796 | | |

Quadro 3.5 – Melhores atributos.

| **Nome do Recurso** | **Descrição** |
|---|---|
| Flow Duration | Duração do fluxo em microssegundos. |
| TotLen Fwd Pkts | Tamanho total do pacote na direção direta. |
| Fwd Pkt Len Mean | Tamanho médio do pacote na direção direta. |
| Bwd Pkt Len Min | Tamanho mínimo do pacote na direção inversa. |
| Bwd Pkt Len Std | Tamanho de desvio padrão do pacote na direção inversa. |
| Flow IAT Mean | Tempo médio entre dois pacotes enviados no fluxo. |
| Flow IAT Std | Desvio padrão do tempo entre dois pacotes enviados no fluxo. |
| Flow IAT Min | Tempo mínimo entre dois pacotes enviados no fluxo. |
| Fwd IAT Mean | Tempo médio entre dois pacotes enviados na direção direta. |
| Fwd IAT Min | Tempo mínimo entre dois pacotes enviados na direção direta. |
| Bwd IAT Mean | Tempo médio entre dois pacotes enviados na direção inversa. |
| Fwd PSH Flags | Número de vezes que o sinalizador PSH foi definido em pacotes que viajam na direção direta (0 para UDP). |
| Fwd Pkts/s | Número de pacotes diretos por segundo. |
| Bwd Pkts/s | Número de pacotes inversos por segundo. |
| SYN Flag Cnt | Número de pacotes com SYN. |
| PSH Flag Cnt | Número de pacotes com PUSH. |
| ACK Flag Cnt | Número de pacotes com ACK. |
| Pkt Size Avg | Tamanho médio do pacote. |
| Subflow Fwd Byts | O número médio de bytes em um subfluxo na direção direta. |
| Init Fwd Win Byts | O número total de bytes enviados na janela inicial na direção direta. |
| Init Bwd Win Byts | O número total de bytes enviados na janela inicial na direção inversa. |
| Active Mean | Tempo médio em que um fluxo estava ativo antes de ficar ocioso. |
| Active Min | Tempo mínimo em que um fluxo estava ativo antes de ficar ocioso. |

Fonte: (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

Para uma melhor avaliação dos modelos propostos, foi executada uma análise dos modelos aplicados a classificação de dados de uma fonte diferente da utilizada no treinamento. Os dados utilizados pertencem a base de dados CSE-CIC-IDS2018 (*Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC)*) com mais informações em (CSE, CIC, 2018). Os testes foram realizados sobre o arquivo da base de dados referente ao dia 22-02-2018, que contém fluxo benigno, *Brute Force-Web*, *Brute Force-XSS* e *SQL Injection*.

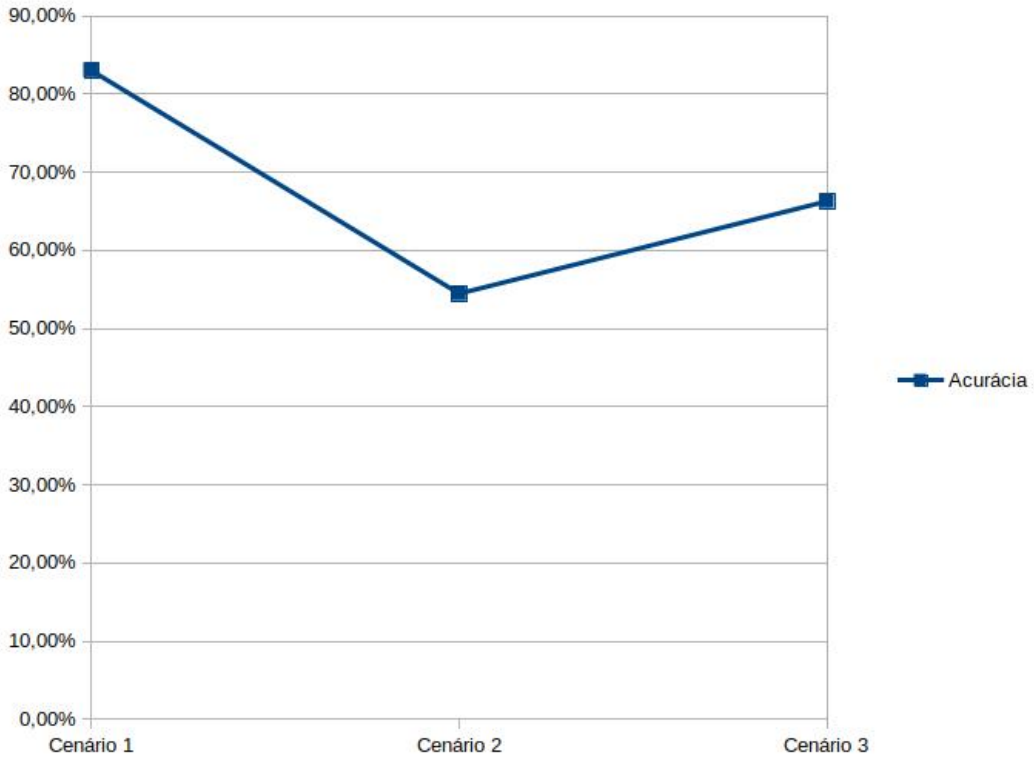
# 4 RESULTADOS E DISCUSSÃO

Descrita a metodologia e o modo de avaliação utilizados, uma série de execuções de treinamento foram realizadas com propósito de gerar dados a serem analisados. Este capítulo aborda as diferentes configurações empregadas no treinamento da rede neural e as características observadas nos distintos cenários propostos, classificação binária quanto para classificação multi-classe. O capítulo conta também com os resultados da análise da validação cruzada dos classificadores, e resultados acerca da aplicação dos classificadores em dados de fontes dados diferentes das utilizadas nos treinamentos.

## 4.1 RESULTADOS DA VALIDAÇÃO CRUZADA

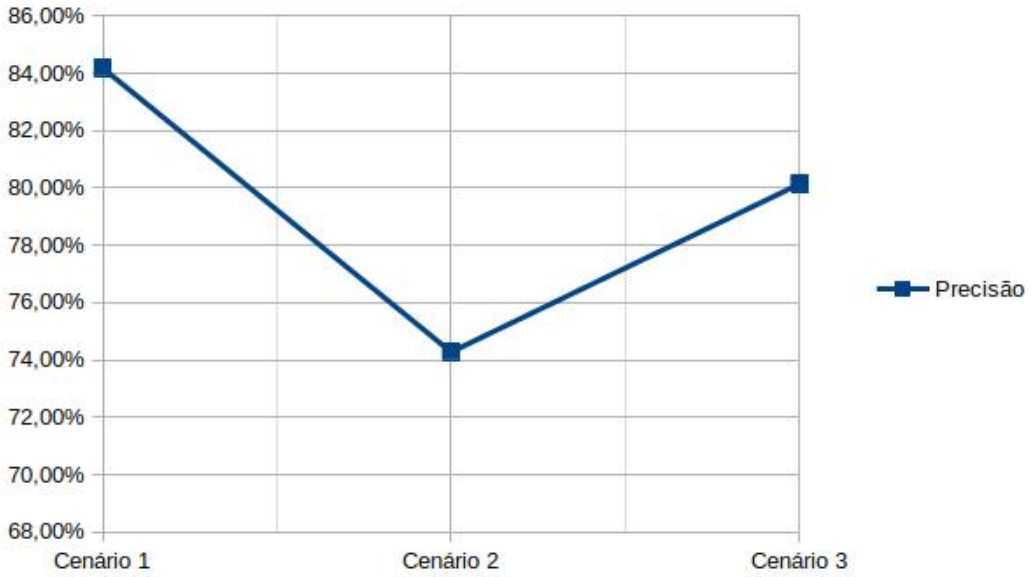
Após o treinamento das RNs profundas com as melhores configurações, foram feitos quatro testes de validação cruzada para cada um dos três classificadores para avaliar a capacidade de generalização dos modelos. O resultado alcançado é demonstrado pelos gráficos nas Figuras 4.1, 4.2 e 4.3. Como pode ser visto na Figura 4.1, o primeiro cenário apresentou a melhor acurácia média nos resultados 83,03% contra 54,47% do classificador do segundo cenário e 66,29% do terceiro classificador.

Figura 4.1 – Gráfico acurácia dos cenários.



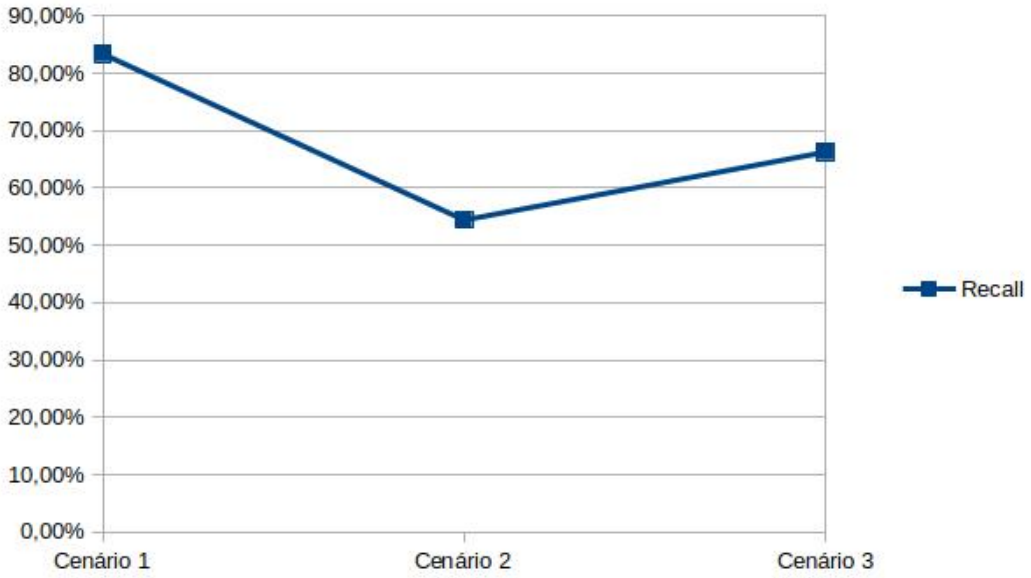
O resultado alcançado na métrica de precisão está disposto na figura 4.2, O gráfico demonstra uma melhor precisão no primeiro cenário 84,18% contra 74,28% do segundo e 80,14% do terceiro cenário.

Figura 4.2 – Gráfico precisão dos cenários.




O resultado alcançado na métrica de recall está disposto na figura 4.3, O gráfico demonstra uma melhor precisão no primeiro cenário 83,37% contra 54,40% do segundo e 66,27% do terceiro cenário.

Figura 4.3 – Gráfico recall dos cenários.



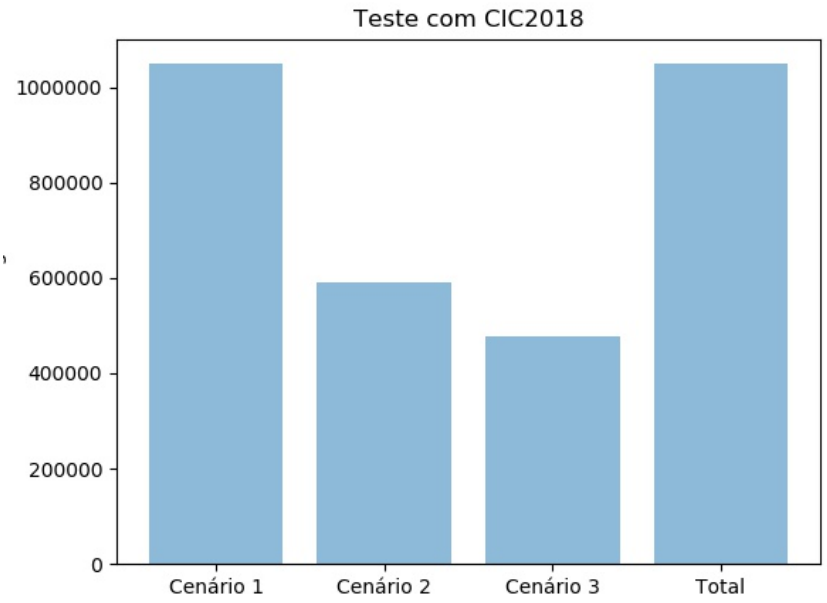

Analisando os gráficos, consegue-se visualizar que o classificador do primeiro cenário (binário) foi o classificador que manteve a precisão, acurácia e recall, mais altos e

próximos um do outro, o que indica que este foi o classificador que se saiu melhor nos testes de validação cruzada.

## 4.2 RESULTADO DA APLICAÇÃO DAS RN'S EM OUTRA BASE DE DADOS

Visando apurar a acurácia das redes treinadas, foram executados testes em registros rotulados obtidos da base de dados do CSE-CIC-IDS2018 (*Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC)*) mais informações em (CSE, CIC, 2018). Foram feitos testes simples para verificar o quanto os algoritmos são capazes de identificar corretamente os registros de uma base diferente da usada em seu treinamento. O resultado obtido como pode ser visualizado na Figura 4.4, demonstra uma eficácia do modelo do primeiro cenário (binário) com 99,96% de acurácia em sua classificação. Para os cenários de classificação multi-classe o modelo do segundo cenário (com proporcionalidade) mostrou-se consideravelmente menos eficiente que o primeiro modelo (binário), com 56,23% de acurácia, e por fim o terceiro modelo (com exclusão) teve o pior resultado com apenas 45,46% de acurácia.

Figura 4.4 – Testes com CIC2018.



# 5 CONSIDERAÇÕES FINAIS

Esse trabalho teve como propósito estudar diferentes estratégias de detecção de anomalias em redes de computadores com aprendizado profundo, utilizou a metodologia de (WANG et al., 2017), que descreve o fluxo de trabalho típico do aprendizado de máquina para redes, os treinamentos foram feitos sobre dados estatísticos retirados de arquivos de log de rede no formato PCAP e salvos em arquivos no formato CSV.

Uma revisão bibliográfica sobre sistemas de detecção de intrusão, anomalias em redes, aprendizado de máquina, aprendizado profundo e trabalhos relacionados, foi feita para fornecer embasamento as técnicas propostas no trabalho. Foram explorados conceitos básicos do aprendizado de máquina e do aprendizado profundo e a relações entre aprendizado de máquina e detecção de anomalias descobertas em trabalhos relacionados.

Na etapa de desenvolvimento, foram projetados e implementados três modelos de classificadores com características distintas em seus treinamentos e tratamentos de dados. Está abordagem visou analisar a capacidade de generalização dos modelos, a fim de fazer uma comparação do desempenho dos classificadores quanto a sua acurácia. Testes foram feitos para escolha dos melhores hiper-parâmetros para o treinamento dos classificadores, também foram feitos testes de validação cruzada para medir o desempenho dos classificadores criados, e por fim, testes com dados de uma base de dados diferente da base usada nos treinamentos foram executados para testar a capacidade de generalização dos classificadores.

Como trabalhos futuros, novos modelos de aprendizado profundo para análise de rede baseados nos modelos propostos neste trabalho podem ser implementados e analisados. Será interessante a execução e análise do sexto passo da metodologia aplicada no desenvolvimento deste trabalho, principalmente para o classificador binário que parece ter certa relevância para detecção de anomalias em redes, mas nada se pode afirmar sem um estudo cientifico especifico sobre essa abordagem.

## REFERÊNCIAS BIBLIOGRÁFICAS


ABNEY, S. **Semisupervised learning for computational linguistics**. [S.l.]: Chapman and Hall/CRC, 2007.

ALVES, F. C. C. et al. Uma revisão sobre as publicações de sistemas de detecção de intrusão. **Revista de Informática Teórica e Aplicada**, v. 23, n. 2, p. 67–99, 2016.

ASHRAF, N.; AHMAD, W.; ASHRAF, R. A comparative study of data mining algorithms for high detection rate in intrusion detection system. **Annals of Emerging Technologies in Computing (AETiC)**, v. 2, n. 1, 2018.

BHUYAN, M. H.; BHATTACHARYYA, D. K.; KALITA, J. K. Network anomaly detection: methods, systems and tools. **Ieee communications surveys & tutorials**, IEEE, v. 16, n. 1, p. 303–336, 2013.

BUCZAK, A. L.; GUVEN, E. A survey of data mining and machine learning methods for cyber security intrusion detection. **IEEE Communications Surveys & Tutorials**, IEEE, v. 18, n. 2, p. 1153–1176, 2015.

BUITINCK, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: **ECML PKDD Workshop: Languages for Data Mining and Machine Learning**. [S.l.: s.n.], 2013. p. 108–122.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. **ACM computing surveys (CSUR)**, ACM, v. 41, n. 3, p. 15, 2009.

CHAPELLE, O.; SCHOLKOPF, B.; ZIEN, A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. **IEEE Transactions on Neural Networks**, IEEE, v. 20, n. 3, p. 542–542, 2009.

CSE, CIC. **A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018)**. 2018. Acesso em 25 nov. 2019. Disponível em: <https://registry.opendata.aws/cse-cic-ids2018/>.

Data Science Academy. **Deep Learning Book**: Home. 2019. Acesso em 01 out. 2019. Disponível em: <http://www.deeplearningbook.com.br/>.

DRAPER-GIL, G. et al. Characterization of encrypted and vpn traffic using time-related. In: **Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)**. [S.l.: s.n.], 2016. p. 407–414.

FADLULLAH, Z. M. et al. State-of-the-art deep learning: Evolving machine intelligence toward tomorrows intelligent network traffic control systems. **IEEE Communications Surveys & Tutorials**, IEEE, v. 19, n. 4, p. 2432–2455, 2017.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **Journal of machine learning research**, v. 3, n. Mar, p. 1157–1182, 2003.

HAAS, A. et al. Um sistema por processamento de fluxos aplicado à análise e monitoramento da rede. Universidade Federal de Santa Maria, 2019.

HAQ, N. F. et al. Application of machine learning approaches in intrusion detection system: a survey. **IJARAI-International Journal of Advanced Research in Artificial Intelligence**, v. 4, n. 3, p. 9–18, 2015.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007.

HYUN, C. M. et al. Deep learning for undersampled mri reconstruction. **Physics in Medicine & Biology**, IOP Publishing, v. 63, n. 13, p. 135007, 2018.

KARATAS, G.; DEMIR, O.; SAHINGOZ, O. K. Deep learning in intrusion detection systems. In: IEEE. **2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)**. [S.l.], 2018. p. 113–116.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

MARZ, N.; WARREN, J. **Big Data: Principles and best practices of scalable real-time data systems**. [S.l.]: New York; Manning Publications Co., 2015.

NGUYEN, T. T.; ARMITAGE, G. A survey of techniques for internet traffic classification using machine learning. **IEEE communications surveys & tutorials**, IEEE, v. 10, n. 4, p. 56–76, 2008.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. **arXiv preprint arXiv:1806.07908**, 2018.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **arXiv preprint arXiv:1710.05941**, 2017.

RYZA, S. et al. **Advanced analytics with spark: patterns for learning from data at scale**. [S.l.]: "O'Reilly Media, Inc.", 2017.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: **ICISSP**. [S.l.: s.n.], 2018. p. 108–116.

SHON, T.; MOON, J. A hybrid machine learning approach to network anomaly detection. **Information Sciences**, Elsevier, v. 177, n. 18, p. 3799–3821, 2007.

THOTTAN, M.; JI, C. Anomaly detection in ip networks. **IEEE Transactions on signal processing**, IEEE, v. 51, n. 8, p. 2191–2204, 2003.

VASCONSELOS, E. W. G. C. C. N. Xxxvi jornadas de atualizaÇÃo em informÁtica. In: FLáVIA C. DELICATO, PAULO F. PIRES, ISMAR FRANGO SILVEIRA, 1., 2017, Porto Alegre/RS. **Anais eletrônicos...** São Paulo: Sociedade Brasileira de Computação - SBC, 2017. Acesso em: 09/2019. Disponível em: <http://www.sbmet.org.br/cbmet2012/pdfs/64ZA.pdf>.

VILLANO, E. G. V. **Classification of logs using Machine Learning Technique**. 2018. Dissertação (Mestrado) — NTNU, 2018.

WANG, M. et al. Machine learning for networking: Workflow, advances and opportunities. **IEEE Network**, IEEE, v. 32, n. 2, p. 92–99, 2017.

YAP, B. W. et al. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In: SPRINGER. **Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)**. [S.l.], 2014. p. 13–22.

YOUNG, S. R. et al. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: ACM. **Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments**. [S.l.], 2015. p. 4.

ZANETTI, S. et al. **ETo RedeNeural Agriambi**. 2012.

ZEILER, M. D. Adadelta: an adaptive learning rate method. **arXiv preprint arXiv:1212.5701**, 2012.

ZHANG, F. **Cross-validation and regression analysis in high-dimensional sparse linear models**. 2011. Tese (Doutorado) — Stanford University, 2011.